Implicit Meshes for Surface Reconstruction

Slobodan Ilic and Pascal Fua
Computer Vision Laboratory
Swiss Federal Institute of Technology
1015 Lausanne, Switzerland
{Slobodan.Ilic, Pascal.Fua}@epfl.ch

Technical Report No: IC/2004/25

Abstract

Deformable 3–D models are used extensively in Computer Graphics and Computer Vision for Visualization, Animation and Modeling. They can be represented either as traditional explicit surfaces, such as triangulated meshes, or as implicit surfaces. Explicit surface representations are widely accepted because they are simple to deform and render. However, for fitting purposes, they suffer from the fact that using them typically involves minimizing a non-differentiable distance function. By contrast, implicit surface representations allow fitting by minimizing a differentiable algebraic distance. However, they have not gained wide acceptance because they are harder to meaningfully deform and render.

Here we propose a method that combines the strength of both approaches. It relies on a technique that can turn a completely arbitrary triangulated mesh, such as one taken from the web, into an implicit surface that closely approximates it and can deform in tandem with it. This allows both automated algorithms to take advantage of the attractive properties of implicit surfaces for fitting purposes and people to use standard deformation tools they feel comfortable for interaction and animation purposes.

We demonstrate the applicability of our technique to modeling the human upper-body, including face, neck, shoulders and ears, from noisy stereo and silhouette data.

1 Introduction

In the world of Computer Vision and Computer Graphics, 3–D objects tend to be modeled as explicit surfaces such as triangulated meshes or parametric surfaces. Because such representations are intuitive and easy to manipulate, they are widely accepted both by researchers and by graphics designers. These representations, however, are not necessarily ideal for fitting surfaces to potentially noisy and incomplete data such as 3–D points produced by laser-scanners and stereo systems or 2–D points from image contours. Fitting typically involves finding the facets that are closest to the 3–D data points or most likely to be silhouette facets, which introduces non-differentiabilities that degrade the convergence properties of most optimizers.

Implicit surfaces, known in the literature as *Blobby Molecules [7], Soft Objects [52] or Metaballs [16]*, have also received substantial attention in both the Computer Graphics and Computer Vision communities. They are well-suited for simulating physically based deformations [47, 49, 32,

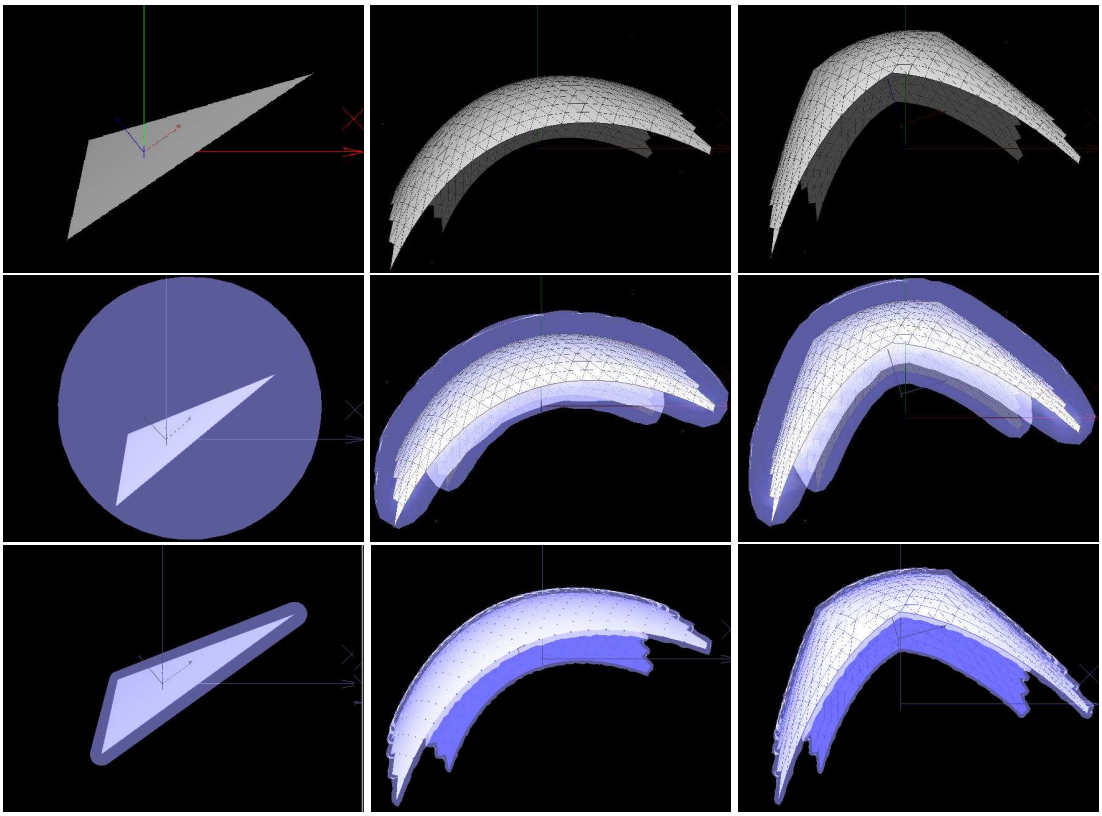


Figure 1: Converting an explicit surface into an implicit mesh. Top row: From left to right, single facet, explicit surface mesh before and after deformation. Middle row: Facet and explicit meshes converted into implicit meshes using spherical primitives. Bottom row: Similar conversion using triangular primitives.

37] and for modeling smooth objects [50, 3, 2]. Because the algebraic distance to an implicit surface is differentiable, they do not suffer from the drawbacks discussed above when it comes to fitting them to 2 and 3–D data [45, 38, 13]. However, they have not gained wide acceptance, in part because they are more difficult to deform and to render than explicit surfaces.

In short, explicit surface representations are well suited for graphics purposes, but less so for fitting and automated modeling. The reverse can be said for implicit surface representations. In this paper, we propose to combine the strengths of both approaches and to avoid their drawbacks by:

1. transforming explicit triangulated surfaces into implicit ones, whose shape closely approximates that of the original triangulations

2. deforming the implicit and the explicit surfaces in tandem for fitting and rendering purposes

As shown in Fig. 1, to create these *implicit meshes*, we attach spherical or triangular volumetric primitives or metaballs to each facet of the explicit mesh. The shape of these primitives depend *only* on the facet geometry. As a result, when a facet deforms, so does the corresponding metaball. To simultaneously control the overall shape of the explicit and implicit meshes, we use Dirichlet Free

Form Deformations [34, 23]. They are a variant of the more traditional Free Form Deformations [42, 12] that lets us control an arbitrarily complex shape using a small number of control points. We will show that this allows us to formulate the fitting problem as a minimization problem with respect to a small number of state variables. Alternatively, because the shape of the implicit mesh strictly is a function of that of the explicit one, we could also have parametrized it in terms of B-splines coefficients [31] or PCA weights [6].

Our contribution is therefore an approach to surface reconstruction that lets us take an explicit surface model of arbitrary complexity and regularity, turn it into an implicit mesh, and take advantage of the attractive properties of implicit surfaces for fitting purposes. Because the implicit surface closely approximates the explicit one and they deform together, the reshaped explicit mesh is also available for rendering and animation. This lets us handle arbitrary triangulations that were not necessarily designed with fitting in mind, such as those that can be found on the web.

In the remainder of the paper, we first briefly review earlier approaches. We then introduce our approach to creating implicit meshes and deforming them. Next we describe our optimization framework. Finally we demonstrate its applicability to the complex task of fitting upper-body model that include head, neck, shoulders, and ears, to stereo and silhouette data.

2 *Related work*

2.1 *Fitting Explicit Surfaces*

Three-dimensional reconstruction of visible surfaces continues to be an important application of Computer Vision and many approaches relying on full 3–D explicit representations, such as 3–D surface meshes [11, 48], parameterized surfaces [44, 30], local surfaces [18], and particle systems [46], have been proposed. In the Computer Graphics world, there has also been a great deal of work on fitting parametric surfaces, such as B-spline patches [31, 17, 27, 33] or subdivision surfaces[15, 28], to 3D data. B-spline patches and subdivision surfaces [10, 14, 29] are widely accepted among graphics designers because they are highly suitable for manual reshaping. They are typically used to reconstruct surfaces from relatively clean laser-scanner data without using a predefined model. Some of these methods can automatically retrieve the structure from unorganized sets of data points [31, 15]. However, when dealing with very noisy and incomplete data such as the stereo disparity maps we use, they are less than ideal. They require an initial polygonalization of the data to create the B-spline patch network, which is hard to obtain in the presence of noise. Fitting parametric models also relies on data parametrization, which means that data points have to be expressed in terms of the model parameters. This requires either projecting them on interactively chosen planes or manually constructed base surfaces [19, 40, 35], or reorganizing them so that they become uniformly distributed, as though they came from a regular grid [27]. Furthermore, tangent plane or G^1 continuity constraints have to be enforced at the boundaries between patches. When dealing with unorganized and noisy data, this may result in an ill-posed constrained least-square problem.

By contrast, Dirichlet Free Form Deformations (DFFDs) [34] let us parametrize an arbitrary mesh, which may be large, irregular and not designed for fitting purposes, in terms of a relatively small number of control points and therefore parameters. In earlier work [23], we took advantage of this property to robustly fit explicit surfaces to noisy stereo data. The control points are usually taken to be on the surface triangulation, with a higher density where the curvature is highest or the shape most likely to change.

As an added bonus, this approach to deforming a surface is intuitive and can be used by a graphics designer to further improve the model if need be. However, because fitting an explicit triangulated surface involves minimizing a non-differentiable objective function, the results are not as accurate as the ones we obtain using the technique proposed in this paper. This will be discussed in more details in Section 6.

2.2 *Fitting Implicit Surfaces*

There has also been sustained interest in the use of volumetric primitives [26, 48, 36] and implicit surface representations [13, 45, 39] for fitting purposes. Most of these methods have been tailored for specific shapes such as the human body. However, more generic methods that rely on implicit algebraic surface splines have also been investigated [2, 1]. These representations are based on both the parametric and implicit nature of B-spline basis functions. However, complex shapes may require very many patches, which would result in a large number of control parameters to optimize if one were to fit such a model to 3–D data.

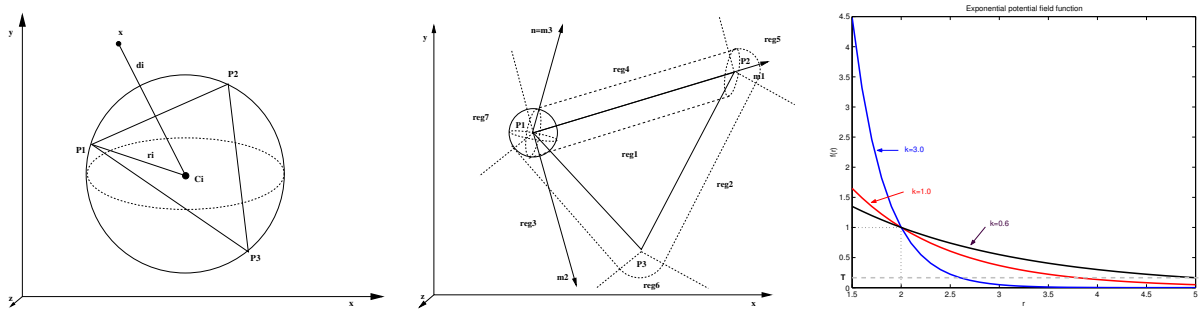
A popular way to deform implicit surfaces is to twist, bend, and taper the space in which the model lives by choosing a suitable warping function [42, 7, 5, 51]. However, these deformations are limited to the surfaces which have both parametric and implicit representations, such as spheres or cylinders. In [4], simple superquadrics are parametrized using conventional FFDs for automatic heart reconstruction and deformation from medical images. Here the FFDs ability to deform parametric surfaces has been exploited, but only to reshape a single primitive. Our proposed implicit meshes coupled with DFFDs go much further by letting us deform completely generic implicit surfaces.

In spirit, our implicit meshes are related to distance surfaces [8]. However, in this work, the problems associated to bulges created by metaballs blending into each other were handled by a convolution mechanism that gives up the algebraic nature of the distance function and makes it impractical for the kind of fitting we perform.

Radial basis functions (RBF) [9, 25, 50] are an interesting alternative to soft objects or metaballs [52, 16]. The shape of the resulting surface, however, is controlled not only by the position of the RBF centers but also by the RBF weights that have no geometric interpretation, which also makes this approach unsuitable for in-tandem deformation of explicit and implicit surface.

2.3 Explicit vs Implicit

In short, both approaches to 3–D modeling have their strengths and weaknesses for the purpose of fitting noisy image-data. Explicit surfaces are easy to deform and to render using well known computer graphics techniques, but are not ideal for fitting purposes because of the non-differentiability of the distance function. Implicit surfaces do not suffer from this problem [45, 39]. However, unless one uses either a single geometric primitive or a set of such primitives attached to some kind of skeleton or complex implicit surface patches, it is relatively difficult to control their shape in an intuitively pleasing way. As a result users such as graphics designers tend to prefer explicit models. To combine the strengths of both approaches, it is therefore important to be able to go back and forth between the two kinds of representations, which is what we propose here.



(a) (b) (c)

Figure 2: Triangular facet enclosed with the (a) Spherical metaball and (b) Triangular metaball, labeled according to the Eqs. 1 and 6. (c) Exponential potential field function, showing how the smooting parameter k controls the range of influence of the primitives and, thus, the amount of smoothing.

3 From Explicit to Implicit Meshes

To create an implicit mesh that can deform in tandem with an explicit one, we define an implicit surface that closely approximates the explicit shape and whose deformations depend only on the motion of the explicit mesh vertices.

To this end, we attach a volumetric primitive, or metaball, to each facet. This can be done in two different ways. The simplest is to use spherical primitives, such as those depicted by the middle row of Fig. 1, which are only adapted to fairly regular and high resolution meshes. A more sophisticated approach requires using the triangular metaballs depicted by the bottom row of Fig. 1, which are more complex but can be used to accurately approximate arbitrarily low-resolution or irregular meshes. We describe these two kinds of metaballs in more detail below.

3.1 Spherical Metaballs

A spherical metaball [24] is created by circumscribing a spherical primitive around a facet in such a way that the sphere center $\mathbf{C_i}$ lies on the facet and corresponds to the center of the circumscribed circle around the facet, as shown in Fig 2(a). For facet Fi, it defines a potential field that can be expressed as:

$$f_i(\mathbf{x}) = exp(-k(d_i(\mathbf{x}) - r_i)) \quad , \tag{1}$$

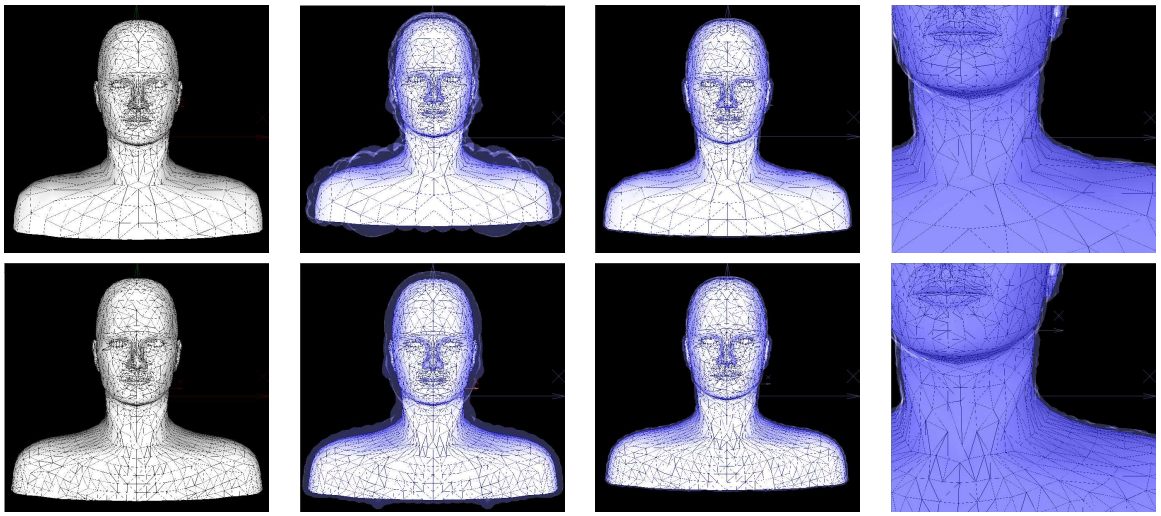
where $\mathbf{x}$ is a 3–D point, $d_i(\mathbf{x})$ is the Euclidean distance of $\mathbf{x}$ to $\mathbf{C_i}$, r_i is the radius of the spherical metaball, and k is a parameter that controls the smoothness of the overall implicit surface and will be discussed below. C_i is determined from:

$$\mathbf{C_i} = \frac{3\mathbf{G_i}(P_1, P_2, P_3) - \mathbf{H_i}(P_1, P_2, P_3)}{2} \quad , \tag{2}$$

where $\mathbf{G_i}$ is facet's center of gravity, and $\mathbf{H_i}$ its orthocenter, both defined by the facet's vertices P_1, P_2 and P_3. We can then write

$$d_i(\mathbf{x}) = \|(\mathbf{x} - \mathbf{C_i})\| \quad , \tag{3}$$

$$r_i = \|(\mathbf{P_1} - \mathbf{C_i})\| \quad . \tag{4}$$



(a) (b) (c) (d)

Figure 3: Conversion of low and high resolution explicit meshes to implicit ones. (a) Low and high resolution meshes. (b) Corresponding implicit surfaces created using spherical metaballs. The volume enclosed by the implicit surfaces is shown in gray. (c) Corresponding implicit surfaces created using triangular metaballs. The enclosed volume is still shown in gray but, for both the low and high resolution meshes, the implicit surface is now so close to the explicit one that it is almost impossible to see at this resolution. (d) Magnified view so that the small difference between the implicit and explicit meshes, which is a function of the d_0 parameters of Eq. 7, becomes visible.

Note that both $\mathbf{C_i}$ and r_i depend only on the positions of the facet vertices.

The implicit mesh is taken to be an isosurface of the sum of all these potential fields. Formally, this isosurface is the set of 3–D points $\mathbf{x} \in R^3$ that satisfy

$$F(\mathbf{x}) = T - \sum_{i=1}^{N} exp(-k(d_i(\mathbf{x}) - r_i)) = 0 \quad , \tag{5}$$

where T is an arbitrarily chosen isovalue. Usually we take T to be one, so that all points on the surface have a zero potential field value, the values smaller than zero are inside and those greater than zero outside.

The isosurface is approximately parallel to the mesh on both sides and encloses the volume shown in gray in the middle row of Fig. 1. Its thickness is a function of the r_i radii of the metaballs attached to the individual facets, and therefore of their sizes. As shown in Fig. 2(c), for values of k greater than one, the exponential drops fast and the individual metaballs have influence only over a relatively short range. As a result, the isosurface closely follows the shape of the metaballs and can be very bumpy if the facets are irregular. For values of k smaller than one, the metaballs tend to blend into each other, which yields a smoother but thicker isosurface. The first is desirable but the second can result in the problems depicted by Fig. 4. In practice, we found that by setting k to values around 1.0 both requirements are fulfilled.

Because the spherical metaballs are circumscribed around the facets their radius r_i depends on the size of the triangle. As shown in the second row of Fig. 1, as long as the explicit mesh is relatively regular

(a) (b) (c) (d)

Figure 4: Relationship between the accuracy of the approximation and the quality of the fitting results. (a) An explicit mesh is approximated using spherical metaballs, which results in an implicit surface of a certain thickness. (b) If the original mesh intersects the data represented by small circles, different sides of the implicit surface may become attracted to the data, resulting in a poor fit. (c,d) Using triangular primitives yields a much thinner implicit surface and a much improved fit.

or high resolution, this yields a valid approximation. However, because large facets produce large primitives, the approximation becomes much less accurate as the facets of the explicit mesh increase in size. When dealing with low resolution irregular meshes, such as the one of Fig. 3(a), elongated facets produce 1 implicit surface that enclose a volume whose thickness can change dramatically, as shown in Fig. 3(b).

Up to a point, that can be remedied by refining the mesh as shown in Fig. 3(a), so that it consists of many smaller size facets and produces the better approximation of Fig. 3(b). However, this results in a substantial computational cost increase. Furthermore, the volume enclosed by the isosurface remains relatively thick, unless the facets are made to be even smaller, which would become prohibitively expensive.

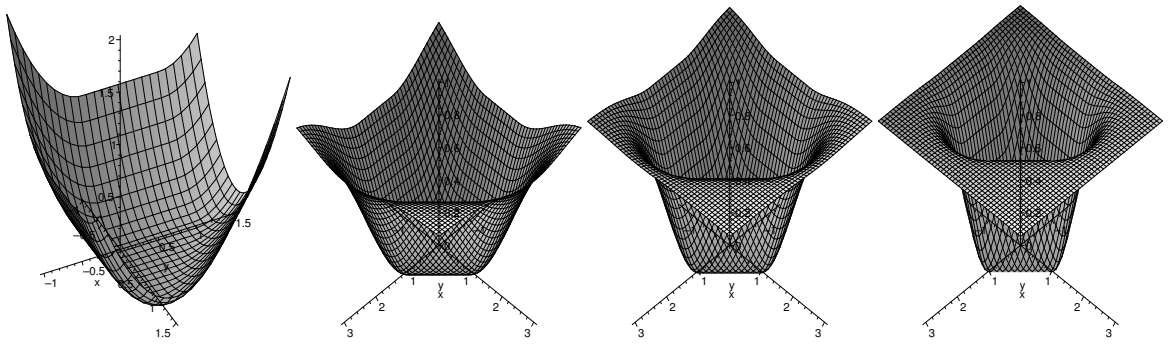
3.2 Triangular Metaballs

To solve these problems, and create implicit surfaces that more closely approximate arbitrary meshes, we can replace the spherical metaballs by *triangular* ones [24], such as those depicted by the bottom row of Fig. 1.

This is done by replacing the Euclidean distance to the facet's center of Eq. 1 by a distance $d(\mathbf{x})$ that more closely approximates the orthogonal distance to the whole facet. We could, of course, take $d(\mathbf{x})$ to simply be the orthogonal distance to the facet plane but that would mean that all points on that plane have a zero distance, no matter how far they are from the facet. Instead, we define $d(\mathbf{x})$ as a piecewise-polynomial function as follows.

For a given facet F_i, we compute the partition of the plane it defines in the seven regions depicted by Fig. 2(b). Given a point $\mathbf{x} \in R^3$, we compute its projection on the facet plane and, depending in which region it falls, we take its distance to F_i to be

$$d(x) = \begin{cases} \left(\frac{\mathbf{n}\bullet(x-P_1)}{\|\mathbf{n}\|}\right)^2 & , \quad \mathbf{x} \in reg1 \\ \frac{\|(\mathbf{x}-\mathbf{P}_i)\bullet\mathbf{e}\|^2}{\|\mathbf{e}\|^2} & , \quad \mathbf{x} \in reg2,\ reg3,\ reg4 \\ \|\mathbf{x}-\mathbf{P}_i\|^2 & , \quad \mathbf{x} \in reg5,\ reg6,\ reg7 \end{cases} \tag{6}$$



(a) (b) (c) (d)

Figure 5: (a) Distance function of Eq. 6 and (b) potential field function of Eq. 7 with different values of parameters $k = \{0.5, 1.0, 2.0\}$, for one facet laing in xy-plane and fixed z coordinate.

When the 3–D point $\mathbf{x}$ projects inside the facet, it falls within $reg1$ and $d(\mathbf{x})$ is simply the squared orthogonal distance to the facet plane, expressed in terms of its normal $\mathbf{n}$ and vertex P_1. If $\mathbf{x}$ projects outside of the facet but in the bands perpendicular to the edges, it falls within regions $reg2$, $reg3$ or $reg4$ and $d(\mathbf{x})$ becomes the squared Euclidean distance from the closest edge passing respectively through $P_i \in \{P_2, P_1, P_1\}$ whose direction is given by vector $\mathbf{e} \in \{P_1P_2, P_1P_3, P_2P_3\}$ respectively. In the remaining cases, the projection falls within regions $reg5$, $reg6$ or $reg7$ and $d(\mathbf{x})$ is taken to be the Euclidean distance to the closest vertex $P_i \in \{P_2, P_3, P_1\}$. Fig. 5(a) depicts $d(x)$ for a standardized facet lying in the $z = 0$ plane with vertices $P_1 = \{0, 0, 0\}$, $P_2 = \{1, 0, 0\}$ and $P_3 = \{0, 1, 0\}$. Note that on the surface of the triangle distance is uniformly zero while along the edges and at the vertices, we find well-behaved parabolas that blend smoothly.

Note that the distance of a point to a facet's edge that appears on the second line of Eq. 6 is the cylindrical distance to that edge. Similarly, the distance to a vertex that appears on the third line of Eq. 6 is the spherical distance to that vertex. Intuitively, a triangular metaball can be understood as being made of two planes, one on each side of the explicit facet, that blend seamlessly with three implicit cylinders whose axes are aligned with the edges at three implicit spheres centered at the vertices. The cylinders and spheres are represented by dotted lines in Fig. 2(b). In Appendix A, we formalize this observation in terms of a matrix representation. In Appendix B, we then use this representation to outline a formal proof that the distance of Eq. 6 is C^1 with respect to the 3–D coordinates of both the vertices and the $\mathbf{x}$ data points.

Finally, the distance function can be incorporated in the same potential field function as the one used for spherical metaballs

$$f_i(\mathbf{x}) = exp(-k(d_i(\mathbf{x}) - d_0^2)) \quad , \tag{7}$$

where d_0 represents the constant thickness of the implicit surface and replaces the variable spherical radii r_i of Eq. 1. Fig. 5(b,c,d) depicts the potential field function of Eq. 7 for different values of smoothing parameter $k \in \{0.5, 1.0, 2.0\}$. Again, the total field is the sum of the individual metaballs fields, which

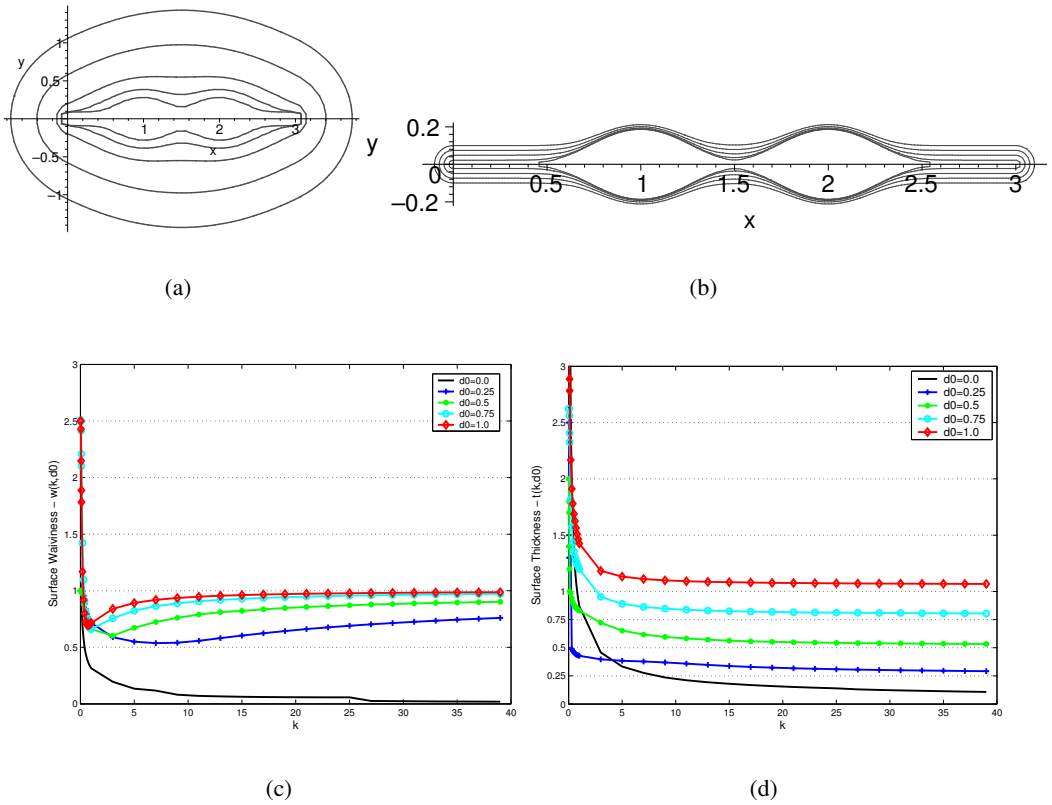


Figure 6: Top row: Zero level-sets of three unit line segments aligned along x-axis for (a) $d_0 = 0.1$ and varing k from 0.1 corresponding to the outer level-set, to 10.0 , corresponding to the inner level-set. (b) $k = 20.0$ and varing d_0 from 0.0 for the inner level-set to 0.1 for the outer level-set. Bottom row: Smoothness and accuracy as a function of the k parameter. (c) Surface waviness and (d) surface thickness as a function of k for different values of d_0.

yields the final expression of the implicit surface as the set of points $\mathbf{x} \in R^3$ such that

$$F(\mathbf{x}) = T - \sum_{i=1}^{N} exp(-k(d_i(\mathbf{x}) - d_0^2) \quad . \tag{8}$$

Note that now, unlike in the case of spherical metaballs, the behavior of the potential field has become independent from facet sizes and mesh resolution. It only depends on the C^1 distance function $d(x)$ and the d_0 and k parameters of Eq. 7 over which we have full control.

As shown in Fig. 3, this lets us approximate arbitrary meshes much more closely, no matter how irregular they may be. As a result, the fitting failure mode depicted by Fig. 4(a,b) can now be overcome. As shown in Fig. 4(c, d), if d_0 is taken to be small enough, it does not really matter to which side of the implicit surface the data is closest because its thickness has become negligible.

In practice, both the d_0 thickness and k smoothing parameters of Eq. 7 influence the smoothness and accuracy of the implicit mesh. To illustrate this, we computed zero level-sets around the three 2–D line segments aligned along the x-axis using a 2–D version of the distance function of Eq. 6. In Fig. 6(a), we

plot the zero level-sets for fixed $d_0 = 0.1$ and varying k, and in Fig. 6(b) for fixed $k = 20.0$ and varying d_0. Increasing d_0 or decreasing k tends to smooth the implicit surfaces, but thickness of the volume they enclose becomes huge, which is highly undesirable as depicted in Fig. 6(a,b). Our goal is therefore to find the best possible compromise between accuracy and smoothness as a function of k and d_0. To quantify the influence of these two parameters, we introduce quantitative measures of smoothness and accuracy that both depend on the thickness of the volume enclosed by the implicit mesh.

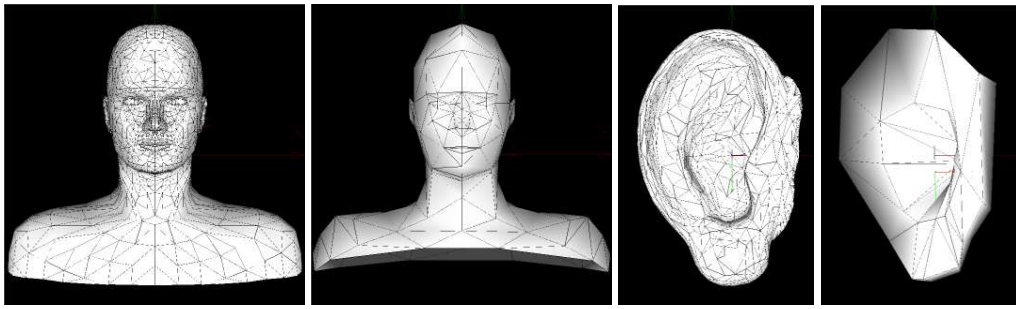
- **Surface waviness =** $w(k, d_0)$: Average ratio of minimal and maximal volume thickness evaluated respectively at the center of gravity of the facets and at their vertices.
- **Surface thickness =** $t(k, d_0)$: Average volume thickness measured at the same locations as those used to evaluate waviness.

In Fig. 6(c,d), we plot these values as a function of k for different values of d_0 in the case of the explicit mesh of Fig. 7(b). The behavior if entirely consistent with the 2–D case depicted by the top row of Fig. 6. For each value of d_0 the waviness of Fig. 6(c) tends towards one when k is increased which means that the bulges disapear. However, the higher the value of d_0, the faster it goes to one. Similarly, as can be seen in Fig. 6(d), the thickness is large for small values of k and assimtoticaly approaches d_0 for huge ones. In practice, we constrain d_0 to be less then 10% of the average edge lenght and seek values of d_0 and k such that

$$w(k, d_0) < 1.0 + w_{max} \ , \tag{9}$$

$$t(k, d_0) < d_0 + t_{max} \ , \tag{10}$$

where w_{max} and t_{max} are two user specified thresholds. Note that for generic models such as the ones of Fig. 7(b), this computation needs only be performed once and the optimal values of k and d_0 stored and reused for all subsequent fits to image data.



(a) (b) (c) (d)

Figure 7: Surface and control triangulations. (a,c) The generic low-resolution triangulation we use of upper-body and the ear modeling. (b,d) A subset of its vertices serve as DFFD control points, as discussed in Section 4.1. They are themselves triangulated to impose the regularization constraints of Section 5.

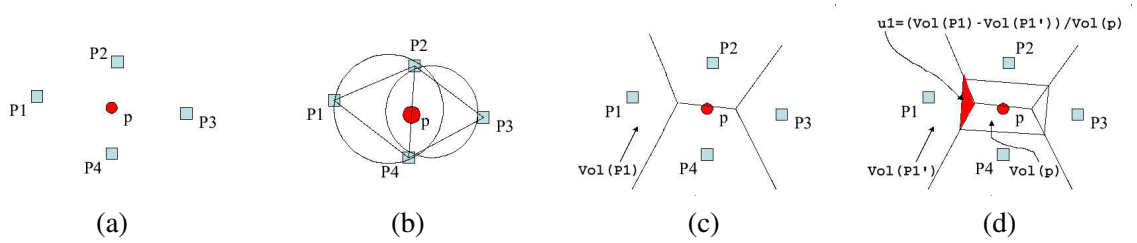


Figure 8: Sibson coordinates. (a) Subset of control points $Q = \{P_1, P_2, P_3, P_4\}$ surrounding mesh vertex p. (b) Delaunay triangulation of the control point set with circumscribed spheres around each Delaunay facet. (c) Corresponding Voronoi diagram. (d) Voronoi diagram for set $Q_p^{'} = \{p, P_1, P_2, P_3, P_4\}$. The Sibson coordinate for control point P_1 is proportional to the area shaded in gray.

4 Deforming Implicit and Explicit Meshes

The shape of the implicit meshes of Section 3 depends only on the position of the 3–D vertices of the corresponding explicit meshes. In theory, for fitting purposes, we could optimize the shape with respect to the x, y, and z coordinates of these vertices. However, because image data is very noisy, we would have to impose a very strong regularization constraint. Instead, we show how we can use Dirichlet Free Form Deformations [34] to parametrize the surfaces, both explicit and implicit, in terms of the positions of a much smaller number of control points.

4.1 Dirichlet Free Form Deformations

In earlier work [23], we showed that Dirichlet Free Form Deformations (DFFDs) [34] constitute an effective way to deform explicit meshes [23]. Unlike other Free Form Deformation methods [42, 12, 22], DFFDs do not require the control points to lay on a regular rectangular grid. This is achieved by replacing the usual rectangular local coordinates by generalized natural neighbor coordinates, also known as Sibson coordinates [43]. This gives us the ability to place control points at arbitrary locations—that is, on the object, inside of it or outside—rather than on a regular lattice, and thus much greater flexibility. In particular, some of the control points can be important feature points that must be controlled in a specific way.

More precisely, given a set of control points such as those of Fig. 7(b,d), every vertex of the explicit mesh of Fig. 7(a,c) is influenced by a subset of those control points. The Sibson coordinates depicted by Fig. 8 quantify those influences and are computed as follows. Let $Q = \{P_1, \ldots, P_N\} \in R^3$ be the set of all control points whose Delaunay triangulation and Voronoi diagram we compute. Let p be a triangulation vertex and $Q_p = \{P_k\}_{1 \leq k \leq N_p}$ be the subset of control points whose circumscribed spheres contain p, as shown in Fig. 8(b). The elements of Q_p are the natural neighbors of p. Their relative influences are obtained by computing the Voronoi diagram of the augmented set $Q'_p = \{p, P_1, \ldots, P_{N_p}\}$ depicted by Fig. 8(d) and taking the Sibson coordinate u_i of vertex P_i to be

$$u_i = \frac{Vol(P_i) - Vol'_p(P_i)}{Vol'_p(p)} \quad , \tag{11}$$

where $Vol(P_i)$ is the volume of the Voronoi cell of P_i in the Voronoi diagram of all the control points and $Vol'_p(P_i)$ and $Vol'_p(p)$ are those in the Voronoi diagram of Q'_p. Note that $\sum_{k=1}^{N_p} u_k = 1$ and

$u_k > 0, \forall k, 1 \leq k \leq N_p$. These coordinates are "natural" in the sense that points in P that are closer to the point p have greater influence on it because the corresponding u_k is larger.

Let the control points in Q_p be displaced from their initial positions by $\Delta P_{k\ ,\ 1 \leq k \leq N_p}$. The position of p becomes

$$p^{new} = p + \sum_{k=1}^{N_p} u_k \Delta P_k \quad . \tag{12}$$

In other words, for any given position of the control points, the overall surface shape is entirely described by the state vector

$$S = [\Delta P_1, \ldots, \Delta P_N] \tag{13}$$

formed by concatenating the displacements of all the control points with respect to their original positions.

4.2 Reparametrizing the Implicit Meshes

To deform an implicit surface created using either spherical or triangular metaballs, it is sufficient to change the parameters that define the shape of the primitives. Let us first consider one single facet and its attached spherical or triangular metaball. A spherical primitive is defined by the distance function of Eq. 1 and the r_i radius, which is a function of the vertex positions. Similarly, a triangular metaball is defined by the distance function of Eq. 6 and the d_0 thickness parameter. In both cases, because the positions of the vertices can be expressed as the weighted linear combination of Eq. 12 of the control points of displacements, the distance functions $d(\mathbf{x})$ of Eqs. 1 and 6 depend not only on $\mathbf{x}$, the 3–D coordinates of the point whose distance is evaluated, but also on the control points. We therefore rewrite the distance function of Eq. 6 and Eq. 3 so to depend on state vector parametres of Eq. 13 like:

$$d(\mathbf{x}) = d(\mathbf{x}, S) \quad , \tag{14}$$

where S is the state vector of Eq. 13.

As a consequence, when considering all the facets together, we can also rewrite the field potential functions F of Eqs. 5 and 8 that define the implicit mesh. When using triangular metaballs, F becomes

$$F(\mathbf{x}, S) = T - \sum_{i=1}^{N} exp(-k(d_i(\mathbf{x}, S) - d_0^2)) \quad , \tag{15}$$

where $\mathbf{x}$ is a point in R^3 and d_i is the distance to facet i defined by Eq. 14. Similarly, for spherical metaballs, we write

$$F(\mathbf{x}, S) = T - \sum_{i=1}^{N} exp(-k(d_i(\mathbf{x}, S)) - r_i(S))) \quad , \tag{16}$$

since r_0 also is a function of the vertex positions.

In this fashion, we have parametrized both the explicit and the implicit surface in terms of the S state vector. As discussed in the following section, this will allow us to deform both representations in tandem to fit the corresponding surface to image data by minimizing a differentiable objective function.

5 Optimization Framework

In this section, we introduce the framework we have developed to fit *generic models* such as the ones of Fig. 7(a,c). Our goal is to deform the surface—without changing the connectivity of its vertices—so that it conforms to the image data, which here comes both, in the form of 3–D point clouds from stereo and 2–D silhouette points from occluding contours. To highlight the capabilities of our implicit meshes, we will perform the fitting using either the implicit or explicit representations in order to compare the results.

5.1 Objective Function for Implicit Meshes

In the case of implicit meshes, we use the image data to write n_{obs} *observation equations* of the form

$$F(\mathbf{x_i}, S) = \epsilon_i \ , \ \ 1 \leq i \leq n_{obs} \ , \tag{17}$$

where F is the field function of either Eq. 15 or 16, $\mathbf{x}_i$ one of the data points, S the state vector of Eq. 13, and ϵ_i the corresponding residual. ϵ_i is the algebraic distance to the implicit mesh and should be as small as possible. Fitting therefore implies minimizing

$$\chi^2 = \mathbf{v}^t W \mathbf{v} \tag{18}$$

where $\mathbf{v} = [\epsilon_1, \ldots, \epsilon_{n_{obs}}]$ is the vector of residuals and W a diagonal weight matrix associated to the observations. In practice, our system must be able to deal with observations coming from different sources. To guarantee that their influences are commensurate, we assign a weight w_{type_i} to each kind of observation, where $type_i$ is the nature of the observation, and minimize

$$\chi^2 = 1/2 \sum w_{type_i} \epsilon_i^2 \ , \tag{19}$$

where the w_{type_i} are chosen so that the contribution to the objective function gradients of all the observations of a particular kind are of similar magnitudes [21].

Because there are both noise and potential gaps in the image data, we found it necessary to introduce a regularization term comparable the one proposed in [4]. Since we start with a generic model, we expect the deformation between the initial shape and the original one to be smooth. This can be effectively enforced by preventing deformations at neighboring vertices of the control mesh to be too different. This is achieved by triangulating the control points as shown in Fig. 7(b) and introducing a deformation energy E_D that approximates the sum of the square of the derivatives of displacements across the control surface. By treating the control triangulation facets as C^0 finite elements, we can write

$$E_D = \Delta_x^T K \Delta_x + \Delta_y^T K \Delta_y + \Delta_z^T K \Delta_z \tag{20}$$

where K is a stiffness matrix and Δ_x, Δ_y and Δ_z are the vectors of the *x, y* and z coordinates of the control vertices' displacements. The term we actually optimize becomes

$$\begin{aligned} E_T &= \chi^2 + \lambda_D E_D \ , \\ &= 1/2 \sum w_{type_i} F(\mathbf{x_i}, S)^2 + \lambda_D E_D \ , \end{aligned} \tag{21}$$

where λ_D is a small positive constant. Note that, because the K stiffness matrix is sparse, adding the E_D regularization term only involves a very small computational overhead when minimizing E_T using

the Levenberg-Marquart algorithm [21] and improves the convergence properties of the algorithm by "convexifying" the objective function.

Because F is differentiable everywhere, so is E_T. Furthermore, this formulation allows us to treat stereo and silhouette data in a similar way, which we discuss below.

5.1.1 Stereo Data

We use either a simple correlation-based technique [20] or a more sophisticated maxflow stereo algorithms [41] to compute potentially noisy clouds of 3–D points. Each one is used to produce one observation of the kind described by Eq. 17. Minimizing the corresponding residuals tends to force the fitted surface to be as close as possible to these points.

The properties of the chosen distance function allow the system to naturally deal with outliers and to converge even from rough initializations or estimates. The smooth shape of the inverted exponential that is used in our field function is responsible for both effects. Because it approaches zero asymptotically, distant data points have an influence, which helps if the initial position is inaccurate, but it is limited. As a result, outliers are naturally ignored because their contribution is dwarfed by that of inliers.

5.1.2 Silhouette Data

For each instance of the state vector $S \in R$, we define the implicit surface

$$L(S) = \left\{ \mathbf{x} \in R^3, F(\mathbf{x}, S) = T \right\} \tag{22}$$

Given the line of sight defined by a silhouette point, let $\mathbf{x}(S)$ be the point along this line of sight where it is tangential to $L(S)$. By definition, $\mathbf{x}(S)$ satisfies two constraints

1. The point is on the surface, therefore $F(\mathbf{x}(S), S) = 0$;
2. The normal to $L(S)$ is perpendicular to the line of sight at $\mathbf{x}(S)$.

We integrate silhouette observations into our framework by performing an initial search along the line of sight to find the point $\mathbf{x}$ that is closest to the model in its current configuration, which by construction satisfies the second constraint. This point is used to add one of the observations described by Eq. 17 and minimizing the corresponding residual tends to enforce the first constraint.

Note that, as the model changes shape, $\mathbf{x}$ must move along the ray to remain the point that is closest to the model. During optimization, this must be taken into account when evaluating the derivatives of the residual because the $\mathbf{x}$ term has now become a function of S. As discussed in Appendix C, this involves evaluating the first and second order derivatives of F.

This approach to taking silhouette information into account does not require us to search for specific facets and imposes no restriction on the topology and complexity of the triangulated model we use. We will see below that such is not the case when using explicit meshes as opposed to implicit ones.

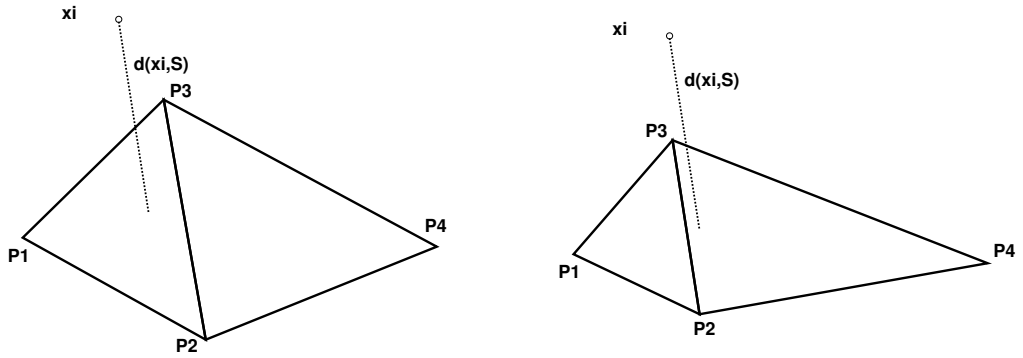


Figure 9: Non-differentiability of the distance function used to fit an explicit mesh to a cloud of 3–D points. (a) A data point $\mathbf{x}_i$ is initially closest to the P_1, P_2, P_3 facet. (b) After a certain number of iterations, the mesh has deformed and $\mathbf{x}_i$ is now closer to the P_2, P_3, P_4 facet. Accounting for this change in the objective function results in non-differentiability.

5.2 Objective Functions for Explicit Meshes

For comparison's sake, we have also implemented an objective function that lets us fit an explicit mesh to the same stereo and silhouette data, but without using the implicit surface formalism we advocate. This objective function is similar to the one of Eq. 22, except for the fact that we must compute differently the ϵ_i residuals that appear in the χ^2 term of Eq. 19. As in Section 5.1, we distinguish between stereo and silhouette data.

5.2.1 Stereo Data

For a 3–D data point $\mathbf{x}_i$, we replace the algebraic distance of Eq. 17 by the orthogonal distance to the "closest" facet

$$\frac{\mathbf{n}(S) \bullet (\mathbf{x}_i - \mathbf{p}(S))}{\|\mathbf{n}(S)\|} \quad , \tag{23}$$

where $\mathbf{n}(S)$ is a vector normal to the facet and $\mathbf{p}(S)$ one of its vertices.

Here we take the closest facet to be one of those that defines a triangular cylinder that contains $\mathbf{x}_i$ and, if no such facet exists, we simply ignore the data point. As shown in Fig. 9, as the optimization progresses and the facets move, the closest facet may change, which introduces non-differentiabilities if taken into account.

5.2.2 Silhouette Data

Given a silhouette point, we look for a triangulation facet that is almost parallel to the line of sight it defines and such that there is a 3–D point along this line for which the distance of Eq. 23 is small. If

such a facet exists, it projects almost edge-on and is therefore likely to produce an occluding contour that goes through the silhouette point.

Let $\mathbf{l}$ be the unit vector that represents the direction of the line of sight. To enforce the silhouette constraint, we search for the facet whose normal is almost perpendicular to $\mathbf{l}$, that is such that $|n \bullet l| \leq \zeta$ where ζ is a small constant, and along which there is a point $\mathbf{x}$ such that the distance of Eq. 23 is smallest. We use this facet and this point to define two residuals

$$\begin{aligned} \epsilon^{silh} &= \frac{\mathbf{n}(S) \bullet (\mathbf{x} - \mathbf{p}(S))}{\|\mathbf{n}(S)\|} \\ \epsilon^{norm} &= \mathbf{l} \bullet \mathbf{n}(S) \end{aligned}$$

whose weighted sum of squares replace the corresponding silhouette terms in Eq. 19.

In addition to the non-differentiabilities that looking for the "closest" facet introduces and that we discussed above, the main problem with this formulation is that these residuals are critically dependent on mesh resolution and regularity. If the facets are irregular or large where the surface slants away from the camera, it will be difficult to find appropriate "silhouette" facets. This results in some of the poor results shown in Section 6 that are corrected by replacing the explicit meshes by implicit ones.

6 Results

To validate our approach, we focus on using stereo and silhouette data because they are complementary sources of information. Stereo works well on textured surfaces facing the camera but fails where the view direction and the surface normal is close to being orthogonal, which is exactly where silhouettes provide robust information.

We first illustrate the behavior of our implicit meshes using synthetic data. We then move on to real data for the upper body, including head, neck, shoulders and ears.

6.1 Synthetic Data

Fig. 10 depicts a synthetic example that simulates a situation in which one must combine stereo and silhouette data to achieve a good result. The left column of the figure depicts the initial shape of an explicit mesh seen from the front and the side. Our goal is to turn this explicit mesh into an implicit one and then to fit it to a surface whose outlines appear as white curved lines. In this example, because the mesh is regular, we can use spherical or triangular primitives indifferently.

We assume that the cameras are in front of the surface we want to model and, therefore, yield stereo data, depicted by the white dots, only on the side facing camera and for the part of the surface that is close to being front-parallel. The middle column of Fig. 10 depicts the result of fitting using this stereo data alone. As could be expected, only the bottom part of the mesh is fitted correctly and the corresponding occluding contours do not match the white outlines. The left column of Fig. 10 depicts the result obtained by combining stereo and silhouette data. The top of the fitted mesh has moved appropriately and the occluding contours it produces are now where they should be. The back of the shape is, of course, still inaccurate since there is neither silhouette nor stereo data to constrain it.

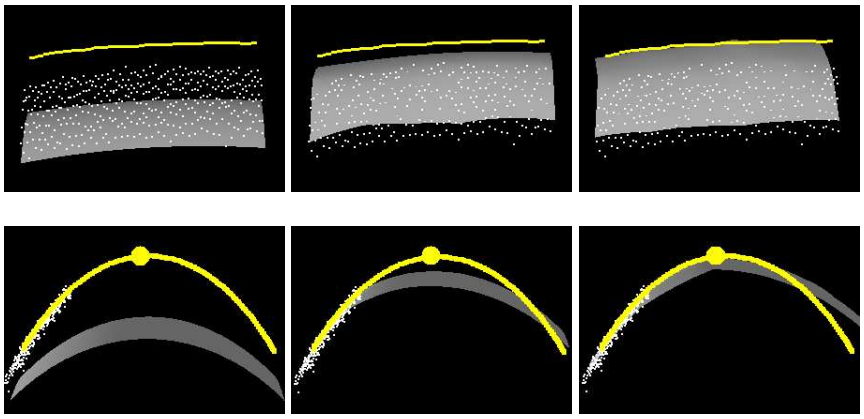


Figure 10: Synthetic example. Left column: Front and side view of an initial mesh shown in light-gray, with occluding contours of a cylindrical 3–D surface to be fitted drawn as white lines. The large white dot in the side view corresponds to the occluding contour in the front view. The smaller white dots represent simulated stereo-data. Middle column: Fitting results using stereo alone. Right column: Fitting results using both stereo and silhouette observations derived from the occluding contour in the front view.

Figure 11: Reconstruction from an uncalibrated video sequence. Top row: 5 of 11 images from a short video sequence with overlaid silhouettes for the neck and shoulders. Middle row: Data clouds extracted from the image using a maxflow graph-cut stereo algorithm, after automated registration. Bottom row: Textured reconstructed models obtained by using a triangular implicit mesh model for the upper body.

Figure 12: Comparing three approaches to fitting an upper body model to the stereo and silhouette data of Fig. 11. Each row depicts the output of one technique, with the true occluding contours shown as white lines. In the outer columns, we show a small scale face and a side view of the fitted models and, in the inner columns, a larger scale view of the neck and shoulder area. Top row: Using explicit surfaces only yields a poor fit of the shoulders. Middle row: Using spherical primitives results in a fit that is better but still inexact at the junction of the neck and shoulders. Bottom row: Using triangular primitives yields the best result.

6.2 Real Data

Here we show a similar behavior, but now using real stereo and silhouette data. In the example of Fig. 11 we use as input an initially uncalibrated 11–frame video sequence in which the camera moves around a static subject. We first used a model-driven bundle-adjustment technique [21] to compute the relative motion and, thus, register the images. We then ran a maxflow graph-cut algorithm [41] to derive disparity maps from consecutive image and produce the clouds of 3–D points depicted by the middle row of the figure. We used snakes to outline the silhouettes shown as white lines. The bottom row of the figure depicts the reconstructed and textured model, obtained using triangular implicit meshes and whose projections align correctly with the silhouettes in all views. This shows that the recovered shape is geometrically correct even at places where the surface slants away from the cameras and, therefore, where stereo fails. Note that these texture-mapped views were generated using standard OpenGL tools to render the explicit surface that was deformed in tandem with the implicit one. In other words, having both kinds of representation simultaneously available at the same time spared us the need to use sophisticated implicit surface rendering techniques.

The results of Fig. 11 were obtained using triangular primitives. For comparison purposes, in Fig. 12, we show the result of fitting the model to the same data using either spherical primitives or directly the explicit surface without taking advantage of the implicit surface formalism proposed in this paper. In all cases, the explicit mesh is parametrized in terms of the same DFFD control points depicted by Fig. 7(b). When *not* using the implicit surfaces, we minimize the objective function of Section 5.2,

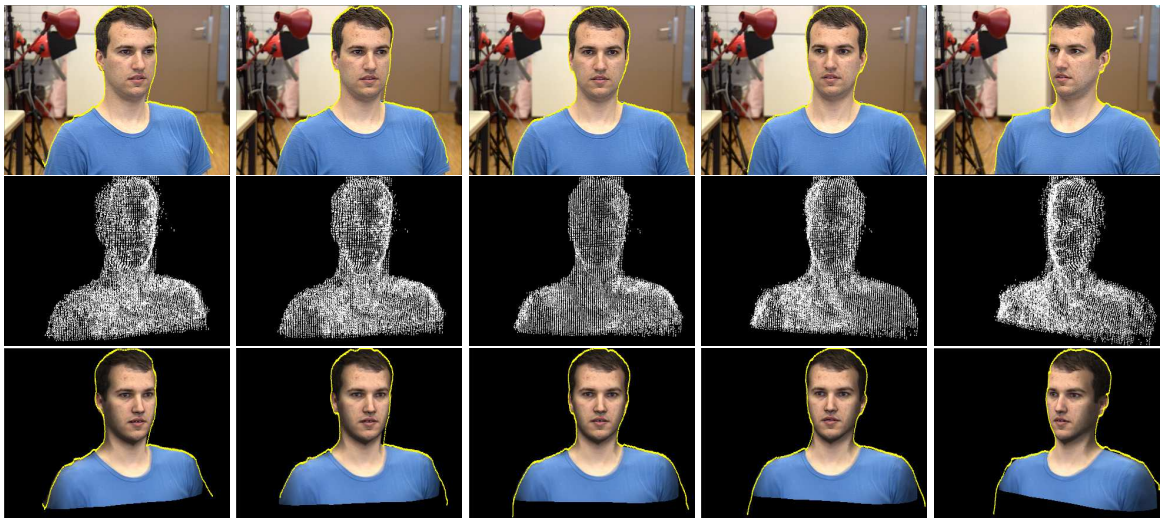


Figure 13: Reconstruction from a shorter uncalibrated video sequence. Top row: 5 of 7 images from a short video sequence with overlaid silhouettes on the head, neck and shoulders. Middle row: Clouds of 3–D points extracted from consecutive image pairs using correlation-based stereo, after automated registration. Bottom row: Textured reconstructed model with triangular implicit mesh model viewed in the same perspective as the original images and with overlaid silhouettes to highlight the quality of the fit.

which unlike those of Section 5.1 is non-differentiable and highly sensitive to the regularity of the mesh facets especially when it comes to handling silhouette information. This results in the fit depicted by the top row of Fig. 12 that is inaccurate in the shoulder area, which is where the silhouettes are the primary source of information. The spherical primitive result shown in the second row is better, but still imprecise at the junction of the neck and shoulders. Close examination of the results show that this is a manifestation of the problem discussed in Section 3.1 and depicted by Fig. 4: Because the facets of the neck and shoulder are of different sizes, the thickness of the implicit surface varies and, at places, the wrong side of it ends up being attracted to the data. As shown in the bottom row of Fig. 12, these problems go away when the spherical primitives are replaced by triangular ones. Note that this is true, even though we fitted the irregular low resolution mesh from the bottom row of Fig. 3 instead of the high resolution one from the top row of Fig. 3 that we used both to directly fit the explicit surface and in conjunction with the spherical metaballs.

Figs. 13 and 14 depict a similar behavior when using a slightly shorter uncalibrated video sequences in which the camera is static and the subject sits on a rotating chair. As before we registered the images using a model-driven bundle-adjustment technique [21]. We then derived the stereo data depicted by the middle row of Fig. 13 using a correlation-based stereo technique [20]. As shown in the top row of Fig. 14, directly fitting the explicit mesh results in an incorrect shape of the shoulder and of the right part of the face. Again, using our implicit meshes with triangular primitives, solves these problems, as shown in the bottom row of the figure.

In our final example, we consider a human ear, whose shape is far more complex than the comparatively simple head models we have used so far. As shown in the top row of Fig. 15, we projected textured light on an ear and acquired a stereo pair of images, which allowed us to compute a fairly dense

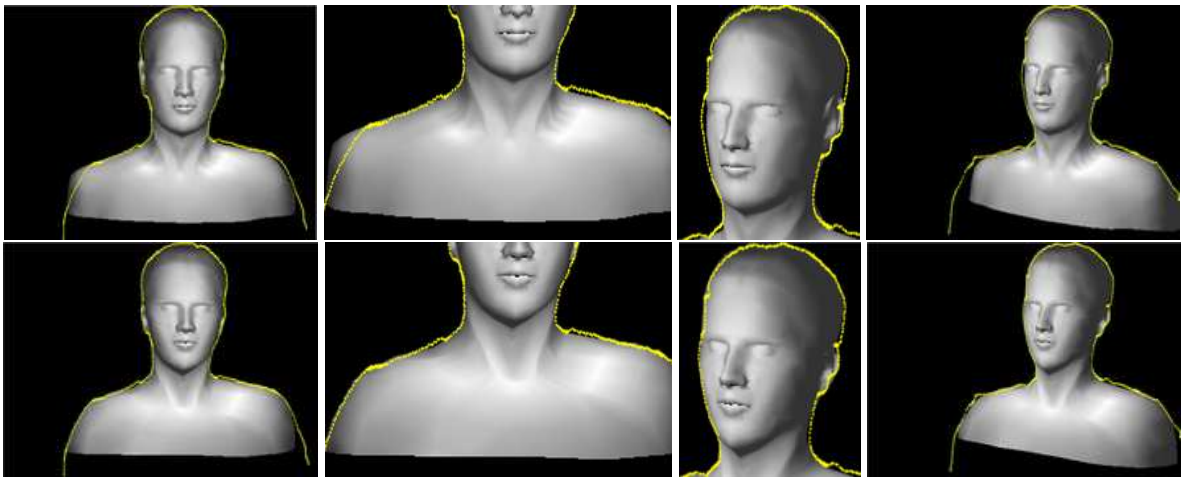


Figure 14: Comparing explicit and implicit approaches to fitting an upper body model to the stereo and silhouette data of Fig. 13. Top row: Directly using explicit surfaces yields a poor fit on the shoulders and the right side of the face, as evidenced by the discrepencies between the surface's occluding contours and the true ones shown as white lines. Bottom row: Using triangular primitives results in a much better correspondence.

disparity map. We then outlined occluding contours in both images and fitted a model we found on the web to this stereo and silhouette data, using the same three techniques as before. Again, only the results obtained using triangular primitives and shown in the bottom row of Fig. 15 correctly line up with the silhouettes and appear realistic.

7 Conclusion

We have presented an approach to combining explicit and implicit surface representations that allows us to take advantage of the strengths of both. To this end, we have developed a technique for creating implicit surfaces from explicit ones by attaching spherical or triangular primitives to their facets and named them implicit meshes. These primitives are defined in such a way that their shape depends only on the 3D location of the mesh vertices, which allows us to simultaneously fit both representations to image data by minimizing a differentiable objective function.

We have chosen to use DFFD control points to parametrize the position of the mesh vertices, which allows us to perform this minimization with respect to a limited number of parameters. Our method, however, is generic and we could also have used another FFD or B-spline based approach for this purpose. We could also have relied on other approaches to reducing the number of parameters, such as Principal Component Analysis of the deformation models [6].

We used the example of upper-body modeling using stereo and silhouette data to demonstrate the power of this approach. The explicit models we used were not tailored for fitting purposes and exhibited both highly irregular facets and a complex topology, none of which had a significant impact on the quality of the fitting.

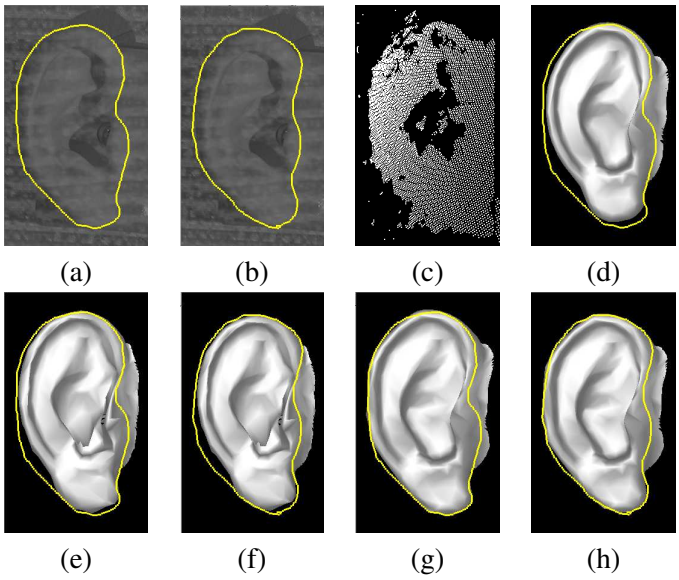


Figure 15: Modeling an ear. (a,b) A stereo pair with overlaid occluding contours. (c) The corresponding cloud of 3D points. (d) Projection of the ear model fitted without using implicit surfaces into one of the images. (e,f) Projections into both images of the model fitted using spherical primitives. (g.h) Similar projections for the model using triangular primitives.

Appendices

A Matrix Representation of the Triangular Metaballs

As discussed in Section 3.2 a triangular metaball, can be understood as being made of two planes, one on each side of the explicit facet, that blend seamlessly with three implicit cylinders whose axes are aligned with the edges and three implicit spheres centered at the vertices.

For a given facet F_i with vertices $\{\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3\}$, let us consider a local coordinate frame attached to it, with its origin at one of the vertices and x-axis aligned with one of the edges. It can be represented in matrix form as

$$M = \begin{bmatrix} \mathbf{m}_1 & \mathbf{m}_2 & \mathbf{m}_3 & \mathbf{P}_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} A & \mathbf{c} \\ \mathbf{0} & 1 \end{bmatrix}, A = \begin{bmatrix} \mathbf{m}_1 & \mathbf{m}_2 & \mathbf{m}_3 \end{bmatrix}_{3x3}, \mathbf{c} = \mathbf{P}_1,$$

where $\mathbf{m}_1, \mathbf{m}_2$ and $\mathbf{m}_3$ are axes of the local coordinate frame, with $\mathbf{m}_1$ being alligned with the x-axis, and $\mathbf{m}_3$ being the facet's normal. A point $\mathbf{x}_l$ in the local coordinate frame is converted to point $\mathbf{x}_g = M\mathbf{x}_l$ in the global coordinate frame. Let $T_r = M^{-1} = \begin{bmatrix} A^T & -A\mathbf{c} \\ \mathbf{0} & 1 \end{bmatrix}$ be the inverse of M. So, a point $\mathbf{x}_g$ in the global coordinate frame is converted to point $\mathbf{x}_l = T_r\mathbf{x}_l$ in the local coordinate frame.

The coordinates of $\mathbf{P_1}$ are a function of the displacements of the control points and therefore of the state vector S of Eq. 13. Let $\mathbf{n}(S) = [n_1, n_2, n_3]$ be the facet normal, which also is a function of S, and let $\mathbf{x}^T = \begin{bmatrix} x_1 & x_2 & x_3 & 1 \end{bmatrix}$ be the 3–D point in space for which the distance is computed. The

distance function of Eq. 6 can now be expressed in matrix form as

$$d(\mathbf{x},\mathbf{S}) = \begin{cases} d_p(\mathbf{x},\mathbf{S}) = \mathbf{x}^T L_1 L_2 \mathbf{x} & , \quad \mathbf{x} \in reg1 \\ d_c(\mathbf{x},\mathbf{S}) = \mathbf{x}^T T_r^T L_c T_r \mathbf{x} & , \quad \mathbf{x} \in reg2,\ reg3,\ reg4 \\ d_s(\mathbf{x},\mathbf{S}) = \mathbf{x}^T T_r^T L_s T_r \mathbf{x} & , \quad \mathbf{x} \in reg5,\ reg6,\ reg7 \end{cases} \tag{A.1}$$

where

$$L_1(S) = \begin{bmatrix} n_1 & 0 & 0 & 0 \\ 0 & n_2 & 0 & 0 \\ 0 & 0 & n_3 & 0 \\ 0 & 0 & 0 & -\mathbf{n}^T \bullet \mathbf{P}_1 \end{bmatrix} \text{ and } L_2(S) = \begin{bmatrix} \mathbf{n}^T & | -\mathbf{n} \bullet P_1 \\ \mathbf{n}^T & | -\mathbf{n} \bullet P_1 \\ \mathbf{n}^T & | -\mathbf{n} \bullet P_1 \\ \mathbf{n}^T & | -\mathbf{n} \bullet P_1 \end{bmatrix}$$

are matrices that define the plane. Similarly, the matrices

$$L_c = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \text{ and } L_s = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

respectively define a cylinder aligned with the x-axis in the global coordinate frame and a sphere centered at the origin of the global coordinate frame.

B Differentiability of the Triangular Metaballs Field Function

We now use the matrix notation introduced above to prove that the distance function $d(\mathbf{x},\mathbf{S})$ of Eq. A.1 is C^1 continue with respect to the x_i coordinates of $\mathbf{x} = [x_1, x_2, x_3]$ and individual parameters s of the state vector $\mathbf{S}$ of Eq. 13.

The distance functions d_p, d_c, and d_s of Eq. A.1 are $C^{\inf}$ over their domains. We simply have to prove that their derivatives are continuous across the boundaries of the regions that appear in Eq. A.1 and are depicted by Fig. 2(b). The first order derivatives over parameter s of the state vector S are

$$\begin{aligned} \frac{\partial d_p(\mathbf{x},\mathbf{S})}{\partial s} &= \mathbf{x}^T \frac{\partial L_1(\mathbf{S})}{\partial s} L_2 \mathbf{x} + \mathbf{x}^T L_1 \frac{\partial L_2(\mathbf{S})}{\partial s} \mathbf{x} \ , \\ \frac{\partial d_c(\mathbf{x},\mathbf{S})}{\partial s} &= 2\mathbf{x}^T T_r^T(\mathbf{S}) L_c \frac{\partial T_r(\mathbf{S})}{\partial s} \mathbf{x} \ , \\ \frac{\partial d_s(\mathbf{x},\mathbf{S})}{\partial s} &= 2\mathbf{x}^T T_r^T(\mathbf{S}) L_s \frac{\partial T_r(\mathbf{S})}{\partial s} \mathbf{x} \ . \end{aligned} \tag{B.2}$$

Similarly the derivatives over coordinate x_i of point $\mathbf{x}$ are

$$\begin{aligned} \frac{\partial d_p(\mathbf{x},\mathbf{S})}{\partial x_i} &= \frac{\partial \mathbf{x}^T}{\partial x_i} L_1 L_2 \mathbf{x} + \mathbf{x}^T L_1 L_2 \frac{\partial \mathbf{x}}{\partial x_i} = 2 L_1 L_2 \frac{\partial \mathbf{x}}{\partial x_i} \ , \\ \frac{\partial d_c(\mathbf{x},\mathbf{S})}{\partial x_i} &= 2\mathbf{x}^T T_r^T(\mathbf{S}) L_c T_r \frac{\partial \mathbf{x}}{\partial x_i} \ , \\ \frac{\partial d_s(\mathbf{x},\mathbf{S})}{\partial x_i} &= 2\mathbf{x}^T T_r^T(\mathbf{S}) L_s T_r \frac{\partial \mathbf{x}}{\partial x_i} \ , \end{aligned}$$

with $\frac{\partial \mathbf{x}}{\partial x_i} = [\delta_{i,1}, \delta_{i,2}, \delta_{i,3}]$ and $\delta_{i,j} = 1$ if $i = j$, 0 otherwise. We now outline the proof that those derivatives are continuous across all three boundary types around vertex $\mathbf{P}_1$. By switching the roles of the vertices, it is easy to see that the same arguments also hold for the other two.

Case 1 *Points along half circles between the regions $reg4$ and $reg7$ that forms the border curve between the cylinder aligned with the x-axis and the sphere centered at $\mathbf{P}_1$.*

This is a trivial case, since at the junction of the sphere and the cylinder, $x_1 = 0$. As a result, d_s and d_c return the same values and have the save derivatives.

Case 2 *Points of the form $\mathbf{x}_1 = \mathbf{P}_1 + \lambda\mathbf{n}, \lambda \in R$ on the line passing through vertex $\mathbf{P}_1$ in the direction of the normal $\mathbf{n}(S) = [n_1, n_2, n_3]$, where the four regions $reg1$, $reg4$, $reg7$, $reg3$ meet.*

We must consider the four corresponding distance functions: d_p, the distance to the plane, d_{c_1} and d_{c_2} the distances to the lines aligned with the $\mathbf{m_1}$ and $\mathbf{m_2}$ axes, and d_s the distance to $\mathbf{P}_1$. By replacing $\mathbf{x}$ by $\mathbf{x}_1 = \mathbf{P}_1 + \lambda\mathbf{n}, \lambda \in R$ in Eqs. B.2 and B.3, it can be easily shown that

$$\frac{\partial d_l(\mathbf{x}, \mathbf{S})}{\partial s} \mid_{\mathbf{x}=\mathbf{x}_1} = 2\lambda \left\|\mathbf{n}\right\|^2 \mathbf{n} \left(\lambda \frac{\partial \mathbf{n}}{\partial s} - \frac{\partial \mathbf{P}_1}{\partial s}\right) \quad , \tag{B.3}$$

$$\frac{\partial d_l(\mathbf{x}, \mathbf{S})}{\partial x_i} \mid_{\mathbf{x}=\mathbf{x}_1} = 2\lambda n_i \left\|\mathbf{n}\right\|^2, n_i \in \{n_1, n_2, n_3\} \quad , \tag{B.4}$$

for d_l being any one of the four distances and for all points on the line.

Case 3 *Points along the plane $\mathbf{x}_2 = \mathbf{x}_1 + \eta\mathbf{m}_1 = \mathbf{P}_1 + \lambda\mathbf{n} + \eta\mathbf{m}_1, \lambda, \mu \in R$ that includes the edge $\mathbf{P}_1\mathbf{P}_2 = \mathbf{m}_1$ that separating region $reg1$ from $reg4$.*

We must consider the two corresponding distance functions: d_p, the distance to the plane and d_{c_1}, the distance to the line aligned with the $\mathbf{m_1}$ axis. As before, by replacing $\mathbf{x}$ by its value in Eqs. B.2 and B.3, it can be shown that

$$\frac{\partial d_l(\mathbf{x}, \mathbf{S})}{\partial s} \mid_{\mathbf{x}=\mathbf{x}_2} = 2\lambda \left\|\mathbf{n}\right\|^2 \left(\frac{\partial \mathbf{n}}{\partial s} (\lambda\mathbf{n} + \eta\mathbf{m}_1) - \mathbf{n}\frac{\partial \mathbf{P}_1}{\partial s}\right) \quad , \tag{B.5}$$

$$\frac{\partial d_l(\mathbf{x}, \mathbf{S})}{\partial x_i} \mid_{\mathbf{x}=\mathbf{x}_2} = 2\lambda n_i \left\|\mathbf{n}\right\|^2, n_i \in \{n_1, n_2, n_3\} \quad , \tag{B.6}$$

for d_l being either one of the two distances and for all points on the plane.

C Computing Silhouette Constrains for Implicit Meshes

Using the notations of Section 5.1, let $\mathbf{l}$ be the unit vector that represents the direction of the line of sight defined by a silhouette point and, as before, let s be an individual parameter of the state vector S of Eq. 13. To minimize the objective function of Eq. 22, we must evaluate $F(\mathbf{x}(s), S)$ and its derivatives, where $\mathbf{x}(s) = (x(s), y(s), z(s))$ is the point along the line of sight such that the value of F is smallest for a given value of s.

We write

$$\frac{dF(x(s), y(s), z(s), S)}{ds} = \frac{\partial F}{\partial x}\frac{\partial x}{\partial s} + \frac{\partial F}{\partial y}\frac{\partial y}{\partial s} + \frac{\partial F}{\partial z}\frac{\partial z}{\partial s} + \frac{\partial F}{\partial s} \quad . \tag{C.7}$$

The $\frac{\partial F}{\partial x}$, $\frac{\partial F}{\partial y}$ and $\frac{\partial F}{\partial z}$ terms can be computed directly from the algebraic expressions of F of Eqs. 5 and 8. The $\frac{\partial x}{\partial s}$, $\frac{\partial y}{\partial s}$ and $\frac{\partial z}{\partial s}$ are derived as follows.

Let $[n_x^1, n_y^1, n_z^1]$ and $[n_x^2, n_y^2, n_z^2]$ be two unit vectors chosen so that, along with $\mathbf{l}$, they form an orthonormal basis. Because $\mathbf{x}(s)$ moves along the line of sight, its motion is always perpendicular to $[n_x^1, n_y^1, n_z^1]$ and $[n_x^2, n_y^2, n_z^2]$. Therefore, we must have

$$0 = n_x^1 \frac{\partial x}{\partial s} + n_y^1 \frac{\partial y}{\partial s} + n_z^1 \frac{\partial y}{\partial s} \ , \tag{C.8}$$

$$0 = n_x^2 \frac{\partial x}{\partial s} + n_y^2 \frac{\partial y}{\partial s} + n_z^2 \frac{\partial y}{\partial s} \ . \tag{C.9}$$

Furthermore, $\mathbf{x}(s)$ is the point at which the line of sight is tangential to the implicit surface, which implies that the gradient of F must be perpendicular to $\mathbf{l}$. Therefore, $\mathbf{x}(s)$ must also satisfy

$$\mathbf{l} \cdot [\frac{\partial F(\mathbf{x}(s), S)}{\partial x}, \frac{\partial F(\mathbf{x}(s), S)}{\partial y}, \frac{\partial F(\mathbf{x}(s), S)}{\partial z}] \ , \ \forall s \in R \ . \tag{C.10}$$

Differentiating Eq. C.10 with respect to s yields

$$\begin{aligned} 0 &= \mathbf{l} \cdot \left[\frac{\partial^2 F}{\partial x \partial x}, \frac{\partial^2 F}{\partial x \partial y}, \frac{\partial^2 F}{\partial x \partial z} \right] \frac{\partial x}{\partial s} \\ &+ \mathbf{l} \cdot \left[\frac{\partial^2 F}{\partial x \partial y}, \frac{\partial^2 F}{\partial y \partial y}, \frac{\partial^2 F}{\partial y \partial z} \right] \frac{\partial y}{\partial s} \\ &+ \mathbf{l} \cdot \left[\frac{\partial^2 F}{\partial x \partial z}, \frac{\partial^2 F}{\partial y \partial z}, \frac{\partial^2 F}{\partial z \partial z} \right] \frac{\partial z}{\partial s} \\ &+ \mathbf{l} \cdot \left[\frac{\partial^2 F}{\partial x \partial s}, \frac{\partial^2 F}{\partial y \partial s}, \frac{\partial^2 F}{\partial y \partial s} \right] \ . \end{aligned} \tag{C.11}$$

Eqs C.8, C.9 and C.11 are three independent linear equations in the three unknowns $\frac{\partial x}{\partial S}$, $\frac{\partial y}{\partial S}$ and $\frac{\partial z}{\partial S}$. Solving them yields the value of these unknowns and, thus, lets us estimate the derivative of Eq. C.7.

References

[1] C. Bajaj. Surface fitting with implicit algebraic surface patches. In *H. Hagen (ed.) Topics in Surface Modeling*, pages 23–52. SIAM Publications, Philadelphia, 1992.

[2] C. Bajaj. The emergence of algebraic curves and surfaces in geometric design. In *R. Martin (ed.) Directions in Geometric Computing*, pages 1–29. Information Geometers Press, Winchester, UK, 1993.

[3] C. Bajaj and I. Ihm. C^1 Smoothing of Polyhedra with Implicit Algebraic Splines. *Computer Graphics, SIGGRAPH Proceedings*, 26(2):79–88, 1992.

[4] E. Bardinet, L.D. Cohen, and N. Ayache. A Parametric Deformable Model to Fit Unstructured 3D Data. *Computer Vision and Image Understanding*, 71(1):39–54, 1998.

[5] A. H. Barr. Local and global deformations of solid primitives. *Computer Graphics, SIGGRAPH Proceedings*, pages 21–30, September 1984.

[6] V. Blanz and T. Vetter. A Morphable Model for The Synthesis of 3–D Faces. In *Computer Graphics, SIGGRAPH Proceedings*, Los Angeles, CA, August 1999.

[7] J. F. Blinn. A Generalization of Algebraic Surface Drawing. *ACM Transactions on Graphics*, 1(3):235–256, 1982.

[8] J. Bloomenthal and K. Shoemake. Convolution Surfaces. *Computer Graphics, SIGGRAPH Proceedings*, 25(4):251–256, 1991.

[9] Jonathan C. Carr, Richard K. Beatson, Jon B. Cherrie, Tim J. Mitchell, W. Richard Fright, Bruce C. McCallum, and Tim R. Evans. Reconstruction and representation of 3d objects with radial basis functions. In *Computer Graphics, SIGGRAPH Proceedings*, volume 2, 2001.

[10] E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer Added Design*, 10:350–355, 1978.

[11] I. Cohen, L. D. Cohen, and N. Ayache. Introducing new deformable surfaces to segment 3D images. In *Conference on Computer Vision and Pattern Recognition*, pages 738–739, 1991.

[12] S. Coquillart. Extended Free-Form Deformation: A sculpturing Tool for 3D Geometric Modeling. *Computer Graphics, SIGGRAPH Proceedings*, 24(4):187–196, 1990.

[13] M. Desbrun and M.P. Gascuel. Animating Soft Substances with Implicit Surfaces. *Computer Graphics, SIGGRAPH Proceedings*, pages 287–290, 1995.

[14] D. Doo and M. Sabin. Behaviour of Recursive Division Surfaces Near Extraordinary Points. *Computer Added Design*, 10(6):356–360, 1978.

[15] H. Hoppe et al. Piecewise Smooth Surface Reconstruction. In *Computer Graphics, SIGGRAPH Proceedings*, pages 295–302, 1994.

[16] H. Nishimura et al. Object Modeling by Distribution Function and a Method of Image Generation. In *Journal od papers given at the Electronic Communication Conf.*, 1985.

[17] L. Fang and D. Gossard. Reconstruction of smooth parametric surfaces from unorganized data points. *Curves and Surfaces in Computer Vision and Graphics*, SPIE-1830(3):226–236, 1992.

[18] F. P. Ferrie, J. Lagarde, and P. Whaite. Recovery of Volumetric Object Descriptions from Laser Rangefinder Images. In *European Conference on Computer Vision*, Genoa, Italy, April 1992.

[19] David R. Forsey and Richard H. Bertels. Hierarchical B-spline refinement. *Computer Graphics, SIGGRAPH Proceedings*, pages 205–212, August 1988.

[20] P. Fua. From Multiple Stereo Views to Multiple 3–D Surfaces. *International Journal of Computer Vision*, 24(1):19–35, August 1997.

[21] P. Fua. Regularized Bundle-Adjustment to Model Heads from Image Sequences without Calibration Data. *International Journal of Computer Vision*, 38(2):153–171, July 2000.

[22] W.M. Hsu, J.F. Hugues, and H. Kaufman. Direct manipulation of Free-Form Deformations. *Computer Graphics, SIGGRAPH Proceedings*, 26(2):177–184, 1992.

[23] S. Ilic and P. Fua. Using Dirichlet Free Form Deformation to Fit Deformable Models to Noisy 3-D Data. In *European Conference on Computer Vision*, Copenhagen, Denmark, May 2002.

[24] S. Ilic and P. Fua. Generic Deformable Implicit Mesh Models for Automated Reconstruction. In *ICCV workshop on Higher-Level Knowledge in 3D Modelling and Motion Analysis*, Nice, France, October 2003.

[25] W. R. Fright J. C. Carr and R. K. Beatson. Surface Interpolation with Radial Basis Functions for Medical Imaging. *IEEE Transactions on Medical Imaging*, 16(1):96–107, 1997.

[26] I.A. Kakadiaris and D. Metaxas. Model based estimation of 3d human motion with occlusion based on active multi-viewpoint selection. In *Conference on Computer Vision and Pattern Recognition*, San Francisco, CA, June 1996.

[27] V. Krishnamurthy and M. Levoy. Fitting smooth surfaces to dense polygon meshes. In *Computer Graphics, SIGGRAPH Proceedings*, pages 313–324, 1996.

[28] Nathan Litke, Adi Levin, and Peter Schröder. Fitting subdivision surfaces. In *Proceedings of the conference on Visualization '01*, pages 319–324. IEEE Computer Society, 2001.

[29] Charles Loop. Smooth Subdivision Surfaces Based on Triangles. Master thesis, Department of Mathematics, University of Utah, 1987.

[30] D. G. Lowe. Fitting parameterized three-dimensional models to images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(441-450), 1991.

[31] H. Hoppe M. Eck. Automatic reconstruction of B-spline surfaces of arbitrary topological type. In *Computer Graphics, SIGGRAPH Proceedings*, pages 325–334, 1996.

[32] D. Metaxas and D. Terzopoulos. Dynamci Deformations of Solid Primitives and Constraints. *Computer Graphics, SIGGRAPH Proceedings*, 26(2):309–312, 1992.

[33] M. Milroy, Bradley, G. Vickers, and D. Weir. G^1 continuity of B-spline surface patch in reverse engineering. *Computer Added Design*, 27:471–478, 1995.

[34] L. Moccozet and N. Magnenat-Thalmann. Dirichlet Free-Form Deformation and their Application to Hand Simulation. In *Computer Animation*, 1997.

[35] Eben Ostby. Describing free-form 3D surfaces for animation. In *Workshop on Interactive 3D Graphics*, pages 251–258, 1986.

[36] A. Pentland and S. Sclaroff. Closed-form solutions for physically based shape modeling and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:715–729, 1991.

[37] A. Pentland and J. Williams. Good Vibrations: Modal Dynamics for Graphics and Animation. *Computer Graphics, SIGGRAPH Proceedings*, 23(4):215–222, 1989.

[38] R. Plänkers and P. Fua. Articulated Soft Objects for Video-based Body Modeling. In *International Conference on Computer Vision*, pages 394–401, Vancouver, Canada, July 2001.

[39] R. Plänkers and P. Fua. Articulated Soft Objects for Multi-View Shape and Motion Capture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2003.

[40] William T. Reeves90. Simple and Complex Facial Animation: Case Studies. In *State of the Art in Facial Animation*, pages 90–106, 1990.

[41] S. Roy and I.J. Cox. A Maximum-Flow Formulation of the N-camera Stereo Correspondence Problem. In *International Conference on Computer Vision*, pages 492–499, Bombay, India, 1988.

[42] T.W. Sederberg and S.R. Parry. Free-Form Deformation of Solid Geometric Models. *Computer Graphics, SIGGRAPH Proceedings*, 20(4), 1986.

[43] R. Sibson. A vector identity for the Dirichlet Tessellation. In *Math. Proc. Cambridge Philos. Soc.*, pages 151–155, 1980.

[44] E. M. Stokely and S. Y. Wu. Surface parameterization and curvature measurement of arbitrary 3-d objects: five practical methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(8):833–839, August 1992.

[45] S. Sullivan, L. Sandford, and J. Ponce. Using geometric distance fits for 3–d. object modeling and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(12):1183–1196, December 1994.

[46] R. Szeliski and D. Tonnesen. Surface Modeling with Oriented Particle Systems. In *Computer Graphics, SIGGRAPH Proceedings*, volume 26, pages 185–194, July 1992.

[47] D. Terzopoulos, J. Platt, A. Barr, and K. Fleicher. Elastically Deformable Models. *Computer Graphics, SIGGRAPH Proceedings*, 21(4):205–214, 1987.

[48] D. Terzopoulos and M. Vasilescu. Sampling and reconstruction with adaptive meshes. In *Conference on Computer Vision and Pattern Recognition*, pages 70–75, 1991.

[49] D. Terzopoulos and A. Witkin. Physically Based Model with Rigid and Deformable Components. *IEEE Computer Graphics and Applications*, 8(6):41–51, 1988.

[50] G. Turk and J. F. O'Brien. Shape transformation using variational implicit surfaces. *Computer Graphics, SIGGRAPH Proceedings*, pages 335–342, August 1999.

[51] B. Wyvill and K. van Overveld. Warping as a modelling tool for csg/implicit models. In *Shape Modelling Conference, University of Aizu, Japan*, pages 205–214. IEEE Society Computer Press ISBN0-8186-7867-4, March 1997. inivited.

[52] G. Wyvill and B. Wyvill. Data Structure for Soft Objects. *The Visual Computer*, pages 2(4)227–234, February 1986.